Remarks

Reconsideration of the above referenced application in view of the enclosed amendment and remarks is requested. Claims 1, 7, 10, 17 and 19-22 have been amended. Existing claims 1 to 22 remain in the application.

ARGUMENT

§112 Rejections:

Claim 2 is rejected under 35 U.S.C. § 112, second paragraph as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. This rejection is respectfully traversed and Claim 2 is believed allowable based on the foregoing and following discussion.

The Examiner asserts that a "virtual machine extension mode" is unclear. Applicants direct the Examiner to look at page 8, paragraph [0020] of the Specification, as originally filed. Differences in existing systems and virtualization technology systems are described as follows:

> "Virtualization technology enables multiple virtual machines to execute simultaneously on the same platform. Some existing systems allow virtual machines to run on non-virtualized platforms with the help from virtualization software. In that case, a virtual machine monitor (VMM) takes care of many aspects of the system, including managing execution the virtual machines. As more virtual machines are added to the platform, overall system performance may suffer because of overhead required in switching between/among virtual machines and maintaining control over each VM. At some point more VMs cannot be added because the machine runs too slowly for desired tasks. In platforms with virtualization technology, a feature in hardware allows the system to switch between contexts faster. This capability provides hardware mechanisms to virtualization software." [Emphasis added]

The Specification also describes the dual mode properties of virtualization technology and describes the virtual machine extension mode (VMX) mode at least on page 6, paragraph [0016]:

> "In order to support virtualization, processors on Intel Architecture platforms may have support for Virtual Machine Extension (VMX) mode. Virtual Machine Extensions (VMX) allows creation of one or more Virtual Machines. VMX mode allows a processor executing in the context of a guest VM to switch control to the VMM based on certain operations performed by the guest VM software. A guest VM always runs in the VMX

> mode. The processor may use a data structure called Virtual Machine Control Structure (VMCS) 107 to determine the conditions under which the control must be transferred to the monitor 105. The criteria could be that whenever a guest VM requests access to certain memory addresses, I/O ports, or registers, the control will automatically switch to the monitor. While executing the guest software code, if the processor realizes that an instruction requires shift of control to the monitor, it generates a VM_exit event. VM_Exit is a condition which causes the processor to suspend execution of the guest software code and transfer control to the monitor."

It will be understood by one of ordinary skill in the art that VMX mode is available on a processor having virtualization technology at a hardware level. While it was believed to be inherent in the claims because of the reference to two processor modes, the independent claims are amended to explicitly recite that the processor comprises a hardware VT enabled platform. In a VT enabled platform, two processing modes exist. Guest VMs operate in a mode called VMX mode. However, a second mode also exists for executing the VMM. Systems without VT used interrupts to switch execution to/from a VM to a VMM, and/or time slicing by the VMM to control which VM was to run at a given moment. As discussed in the specification, as more virtual machines are added to a platform, overall system performance may suffer because of overhead required in switching between/among virtual machines and maintaining control over each VM. A platform with a VT architecture reduces the overhead because a control structure VMCS defines when context should automatically switch from VMX mode to a VMM mode. Interrupts are not used, but a chipset extension is implemented in the hardware.

As it is clearly described that the processor is to have two distinct modes, one of which is a VMX mode to run the VMs, it is believed that Claim 2 is definite, as originally filed. Therefore this rejection should be withdrawn.

§101 Rejections:

Claims 17-22 are rejected under 35 U.S.C. § 101 as not being directed to statutory subject matter. This rejection is respectfully traversed and Claims 17-22 are believed allowable based on the above amendments and foregoing and following discussion.

Claims 17-22 have been amended to recite a *machine accessible storage medium having instructions stored thereon*. As is well understood, the inventor is entitled to describe supporting information sufficient to enable the invention. The description of machine accessible medium in

the specification is merely an example of possible media in existence. However, it is the claims that define the metes and bounds of the invention. It will be clear to one of ordinary skill in the art that a machine accessible storage medium having instruction stored thereon limits the medium to tangible *storage* media, as permitted by law. *In re Beauregard* (citations omitted) clearly holds that computer readable (e.g., machine accessible) storage media are articles of manufacture and therefore statutory. Thus, this rejection should be withdrawn.

§ 102 Rejections:

Claims 1, 3-7, 10, 13-17 and 19-22 are rejected under 35 U.S.C. § 102(e) as being anticipated by U.S. Patent Application Publication 2005/0076324 to Lowell et al. (hereinafter, "Lowell"). this rejection is respectfully traversed and Claims 1, 3-7, 10, 13-17 and 19-22 are believed allowable based on the foregoing and following discussion.

Independent Claims 1, 7, 10 and 17 are amended to more clearly recite that the processor comprises a hardware virtualization architecture. It is described in the Specification as originally filed that "[i]n platforms with **virtualization technology**, a feature in hardware allows the system to switch between contexts faster. This capability provides hardware mechanisms to virtualization software." (See paragraph [0020].) At least in paragraph [0026] the context mode switches between the two processor modes (or contexts) using a *VM_exit* and *resume_VM* commands/traps are described as

> "[i]n systems with **virtualization technology**, this VM_exit may change the mode of the processor. In this embodiment, the switch allows the RSM 205 to take control of the computing platform. The RSM 205 executes the code for the runtime service 201 and places the results of the call into a shared memory portion (not shown). The RSM then switches back to VMX mode and lets the processor execute resume_VM execution." [emphasis added]

Lowell teaches a system where interrupt traps are forced when certain memory accesses are attempted. In Lowell, Para. [0021], it is described that a user assigns different I/O devices to different OS instances via a GUI connected to the VMM. The VMM causes the CPU to trap on addresses of potential interest. Once a trap occurs, a software handler causes the VMM to execute and handle the device I/O or memory access. The VMM must save all context, execute and then chooses to which VM operation should be returned. This scheme is clearly

implemented in software. In the specification as originally filed, at least in paragraph [0020], Applicants discuss that there are disadvantages to these software implementations:

> "Some existing systems allow virtual machines to run on non-virtualized platforms with the help from virtualization software. In that case, a virtual machine monitor (VMM) takes care of many aspects of the system, including managing execution the virtual machines. As more virtual machines are added to the platform, overall system performance may suffer because of overhead required in switching between/among virtual machines and maintaining control over each VM. At some point more VMs cannot be added because the machine runs too slowly for desired tasks."

In contrast to Lowell, the claimed invention requires the processor to comprise virtualization technology having two processor modes. The VM(s) runs in a first processor mode and the VMM is to run in a second processor mode. Lowell describes a processor having only one mode. Lowell's VMM controls the operation of the VMs, but the processor has only one mode and the VMM controls all of the overhead for switching execution between the various hardware partitions. Further, Lowell allows an OS running in a VM to maintain control over some I/O devices. Lowell teach a system that uses trapping of memory accesses via interrupt handlers (see Fig. 5).

The Examiner asserts that Lowell teach "*a monitor to run in a second processor mode* at Para. 6. However, Para. 6 teach only that two operating systems may run in two separate hardware partitions. The hardware partitioning is not literal, and does not cause two processor modes. Lowell refers to hardware partitioning to dedicate some hardware devices to be used exclusively for a single OS. Further, Lowell does not teach that the OS running in a VM is in a separate processor mode from the VMM, but merely teaches that the VMM controls each OS access to the hardware, and may control hardware that is shared among the OS's. Lowell's hardware partition is not controlled by hardware context switching, but solely by the VMM.

In contrast, for instance in Claim 1, it is required that the processor comprises hardware virtualization architecture, which is not suggested by Lowell. It is also required that a virtual management control structure is defined to authorize access to the system resources. Moreover, Claim 1 requires a means for *automatically switching from the first processor mode to the second processor mode based on an attempted access of system resources defined in the VMCS.* Lowell does not automatically switch processor modes, as only one mode is shown, based on an

attempted access of a system resource. Lowell teach only that an interrupt causes the VMM to take control when a memory address of potential interest is accessed. As described in the Specification, system resources are not merely memory addresses. For instance, in paragraph [0021] it is described that "The VMM communicates with the actual hardware driver and passes the appropriate data to the VM. The VMM acts as a central unit to take care of all resource requests for all VMs. Controlled accesses may also include operating the keyboard and mouse which are shared among VMs, or even accessing memory." Paragraph [0027] describes how the context switch automatically transfers control to the second processor mode when a runtime service is called by a VM. Thus, it will be understood that the term "system resource" as recited in the claim does not mean merely a virtual memory address interrupt trap. Lowell, in Para 22, describes that

> "When a virtual address is accessed for which there is not a valid translation in the TLB, the appropriate PTE is read from the current page table, and then loaded into the TLB. This fill operation may be performed by a TLB miss handler in the VMM (on some platforms and architectures), or by firmware (in other architectures), or by PALcode (in the HP Alpha Architecture), or by the hardware (in still other architectures). By managing the CPU's virtual memory translation, the VMM can cause the CPU to trap to the VMM on addresses of potential interest."

Thus, it will be understood by those of skill in the art that Lowell merely traps addresses with invalid translations. This is not the same as automatically switching control to a second processor mode.

As for Claims 3, 13 and 19, the Examiner asserts that Lowell teaches that the monitor may be a runtime services monitor. The Examiner also asserts that the Lowell teaches that the VMM places the results of the runtime services execution in a shared memory accessible to the VM (at Para 33). In fact, Lowell teach only that the "VMM may also make a copy of the hardware description table, modify the copy to reveal or conceal hardware, and present the modified copy to the booting as instance." Lowell teach that a hardware description table may be presented to the OS running in a VM. However, Applicants' claimed invention requires the RSM to execute, in the second processor mode, and then place the results of the service in shared memory. Lowell does not teach executing a runtime service in the VMM, in a second processor

mode, nor does Lowell teach that the results of this service are place in shared memory available to the VM.

As for Claims 4, 15, and 21, while Lowell teaches two OS instances, Lowell fails to teach or suggest that the VMs run in a first processor mode and that the VMM runs in a second processor mode, as recited in the parent claims, wherein the two processor modes are tied to the hardware virtualization technology capabilities of the processor. Moreover, Claims 4, 15 and 21 are believed allowable as being dependent on an allowable base claim.

Claims 5, 6, 8, 14 and 20, are believed allowable as being dependent on an allowable base claim.

Claim 7 requires that the system resources are not accessible to the VM, wherein an attempt to access at least one system resource by a virtual machine automatically switches the processor mode of operation from the first processor mode to the second processor mode and switches execution control to the monitor. As discussed above, Lowell does not teach a hardware virtualization capable processor, nor does Lowell teach automatic context switching based on an attempted access of a system resource.

As for Claims 10 and 17, the Examiner asserts again that Lowell teach two processor modes, which is not the case, as discussed above. Moreover, Lowell does not teach a hardware virtualization technology capable processor that automatically switches context from a virtual machine execution mode to a monitor mode or switching back to a virtual machine execution mode. Lowell's processor has one mode, as legacy processors, and executes the VMM in the same mode as the VM – just switches which process is executing on the processor, not switching execution *modes*.

As for Claims 16 and 22, the Examiner asserts that Lowell teach that the monitor is a runtime service monitor. In fact, Lowell only teach trapping (causing an interrupt) when a memory access translation fails. In contrast, Applicants' claims require that the monitor has a runtime services monitor component. A runtime service is a term well known term of art, especially as pertains to an extensible firmware interface (EFI) architecture, and is described in the specification, at least in paragraph [0005], as comprising "a set of functions that are available to the OS post-boot. Since the code and data that comprise the runtime services are loaded in the OS address space, the runtime services are open to compromise from malicious programs and

hence protection from tampering, corruption or being overwritten is required." Embodiments of Applicants' claimed invention protect the runtime services from malicious tampering because the runtime services are controlled by the VMM. When a VM requests a service to be executed, context is automatically switched to the monitor mode and results of the service are returned to the VM in shared memory. Thus, the VM cannot maliciously modify the service. Lowell fails to teach or suggest this type of operation in a hardware virtualization capable processor. Thus, Lowell fails to teach or suggest each and every element of the claims.

§103 Rejections:

Claims 2, 8, 9, 11, 12 and 18 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Lowell. This rejection is respectfully traversed, and Claims 2, 8, 9, 11, 12 and 18 are believed allowable based on the foregoing and following discussion.

As discussed above, Lowell fail to teach a processor having two modes of execution. The Examiner asserts that Lowell may run on any type of processor. However, combining the teachings of Lowell with the claimed virtualization technology processor will not result in Applicants' invention. Further, in order to assert obviousness, the operation of a virtualization technology capable processor must have been known by one of ordinary skill in the art, at the time of the invention. The Examiner fails to show that this type of processor was known, or would be obvious to combine with the teachings of Lowell.

Further, Lowell does not teach a means for automatically switching from the first processor mode (VMX mode) to a second processor mode. Even if Lowell was to be combined with a virtualization technology processor, the resulting system would still merely use interrupts to trap untranslatable memory addresses. This operation is not the same type of operation as recited in Claim 2.

As for Claims 8, 11 and 18, Lowell teaches only causing interrupts when certain memory is accessed. Lowell does not teach that the memory access is a request to run a runtime service consisting of code or data. Lowell makes no suggestion that memory can be linked to execution of a service, or that accessing the service will automatically cause a context change in processor mode. It would not have been obvious to modify Lowell to create a system that operates as in the recited Claims. Even if a memory location in Lowell was to point to a runtime service,

access of the memory location would generate an interrupt and not automatically switch the processor mode to monitor mode from virtual machine execution mode, in response to the attempted access, because Lowell teach only one mode.

As for Claims 9 and 12, there is no suggestion in Lowell that the memory address contains a function pointer. Even in the memory access in Lowell was to contain a function pointer, Lowell teach that an interrupt is to be generated, and not automatically switch the processor mode to monitor mode from virtual machine execution mode, in response to the attempted access, because Lowell teach only one mode.

The Examiner fails to understand the difference in operation of a processor comprising hardware virtualization technology architecture and the automatic context switching from one mode to another. Lowell teach generating interrupts in a legacy processor having only one mode. There is no suggestion in Lowell that the memory accesses can be related to runtime services, or that the processor has more than one mode. Thus, Lowell fails to teach each and every limitation recited in the independent claims. Further, there is no suggestion at all that Lowell may be implemented on a processor having two modes of execution, or that the mode switches are controlled by a VMCS. Even if Lowell were implemented on a processor having such an architecture, Lowell would generate interrupts and not control execution modes through the VMCS. Therefore combining Lowell with common knowledge (at the time of filing) does not provide *prima facie* evidence of obviousness. All pending claims are therefore believed to be allowable.

CONCLUSION

In view of the foregoing, Claims 1 to 22 are all in condition for allowance. If the Examiner has any questions, the Examiner is invited to contact the undersigned at (703) 633-6845. Early issuance of Notice of Allowance is respectfully requested. Please charge any shortage of fees in connection with the filing of this paper, including extension of time fees, to Deposit Account 50-0221 and please credit any excess fees to such account.

Respectfully submitted,

Dated: 21 May 2008

/ Joni D Stutman-Horn /
Joni D. Stutman-Horn, Reg. No. 42,173
Patent Attorney
Intel Corporation
(703) 633-6845

Intel Corporation
c/o Intellevate, LLC
P.O. Box 52050
Minneapolis, MN 55402